


Article

Reducing Uncertainty and Increasing Confidence in Unsupervised Learning

Nicholas Christakis ^{1,2,*} and Dimitris Drikakis ^{1,*} ¹ Institute for Advanced Modelling and Simulation, University of Nicosia, Nicosia CY-2417, Cyprus² Laboratory of Applied Mathematics, University of Crete, GR-70013 Heraklion, Greece

* Correspondence: nchristakis@tem.uoc.gr (N.C.); drikakis.d@unic.ac.cy (D.D.)

Abstract: This paper presents the development of a novel algorithm for unsupervised learning called RUN-ICON (Reduce UNcertainty and Increase CONfidence). The primary objective of the algorithm is to enhance the reliability and confidence of unsupervised clustering. RUN-ICON leverages the K-means++ method to identify the most frequently occurring dominant centres through multiple repetitions. It distinguishes itself from existing K-means variants by introducing novel metrics, such as the Clustering Dominance Index and Uncertainty, instead of relying solely on the Sum of Squared Errors, for identifying the most dominant clusters. The algorithm exhibits notable characteristics such as robustness, high-quality clustering, automation, and flexibility. Extensive testing on diverse data sets with varying characteristics demonstrates its capability to determine the optimal number of clusters under different scenarios. The algorithm will soon be deployed in real-world scenarios, where it will undergo rigorous testing against data sets based on measurements and simulations, further proving its effectiveness.

Keywords: unsupervised learning; machine learning; artificial intelligence; uncertainty

MSC: 68T01



Citation: Christakis, N.; Drikakis, D. Reducing Uncertainty and Increasing Confidence in Unsupervised Learning. *Mathematics* **2023**, *11*, 3063. <https://doi.org/10.3390/math11143063>

Academic Editor: Faheim Sufi

Received: 9 June 2023

Revised: 5 July 2023

Accepted: 7 July 2023

Published: 11 July 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

1.1. Description of Unsupervised Learning

Unsupervised learning (UL) has been established as a machine learning (ML) approach that eliminates human bias from the analysis [1–3]. We could also take it one step further and claim that UL can be used to reduce *computational bias* or reduce the uncertainty associated with numerical algorithms and processes.

In UL, the user defines the computational groups and not the algorithm itself. UL is better suited for problems of higher complexity in identifying groups. Moreover, it is easier to obtain unlabelled data—experimental, computational or field measurements—than labelled data that require user intervention. However, UL could encompass higher uncertainty in applications because no labels are pre-defined. Several algorithms have been developed, e.g., see [4].

1.2. Related Works in the Literature

The two most popular algorithms employed in UL are K-means and hierarchical clustering: (i) K-means is a popular clustering algorithm that groups similar data points [5]. The algorithm selects k centroids and assigns each data point to the nearest centroid. The centroids are then recalculated based on the mean of the points assigned to them, and the process is repeated until convergence. (ii) Hierarchical clustering is another clustering algorithm that groups data points into nested clusters, e.g., [6]. The algorithm treats each data point as a separate cluster and iteratively merges the most similar clusters until all points belong to a single cluster.

More recently, deep supervised learning techniques have been utilised in unsupervised learning to leverage the power of deep neural networks to extract meaningful representations from unlabeled data. Using these techniques in unsupervised learning aims to exploit their ability to capture intricate patterns and hierarchical dependencies within data. Applications include problems such as image processing [7,8], sleep stage classification [9] and mechanical damage detection [10].

1.3. Existing Challenges in the Literature

Despite significant progress in recent years, there are still challenges and limitations associated with UL techniques in various data analysis scenarios, which are discussed below:

- **Difficulty in evaluating performance:** Unlike supervised learning algorithms, where the performance can be easily evaluated based on the accuracy of the predicted output, evaluating the performance of unsupervised learning algorithms can be challenging. There is no clear measure of success or failure in unsupervised learning, and it often requires human interpretation to determine the usefulness of the learned patterns [11].
- **Limited applicability:** UL algorithms are typically used for data exploration and pattern discovery, and their applicability is often limited to certain data types. For example, clustering algorithms such as K-means are suitable for data with clear groupings but may not be useful for more complex relationships [12].
- **Lack of interpretability:** UL algorithms often produce models difficult to interpret, making it challenging to understand the underlying patterns and relationships in the data. This lack of interpretability can limit the usefulness of UL algorithms in certain applications, such as in the medical field, where understanding the reasons behind the predictions is critical [13].
- **Vulnerability to noise:** UL algorithms can be sensitive to noise in the data, affecting the learned patterns' accuracy. For example, in clustering algorithms, outliers can distort the boundaries between clusters and affect the accuracy of the clustering results [14].
- **Sensitivity to the initial selection of centroids:** K-means algorithms are known to be sensitive to this. The initial choice of centroids can significantly impact the final clustering results. Additionally, K-means assumes that the clusters in the data have isotropic shapes and sizes, which may not hold in many real-world scenarios. These issues may lead to suboptimal solutions, where the algorithm fails to capture the true underlying structure of the data [15].
- **Scalability issues:** While effective in capturing complex relationships among data points, hierarchical clustering techniques suffer from computational intensiveness. As the data set's size increases, hierarchical clustering algorithms' computational requirements become significant. This scalability issue hinders the application of hierarchical clustering to large-scale data sets [16].
- **Sensitivity to data characteristics:** Deep unsupervised techniques, which utilise deep learning architectures for unsupervised learning tasks, can be sensitive to data set characteristics. Factors such as class imbalance, label distribution variations, or data quality differences can influence deep unsupervised algorithms' performance. These characteristics need to be carefully considered and addressed to ensure reliable results [17,18].
- **Dependency on feature learning:** For deep unsupervised techniques to provide reliable results, feature learning plays a crucial role. The algorithms must learn meaningful representations or features from the data to accurately capture the underlying structure. Without effective feature learning, the performance of these algorithms may be compromised, leading to less reliable clustering results [10].

1.4. Current Work and Contributions

This study presents a new algorithm, the Reduced UNcertainty-Increased CONfidence Algorithm (RUN-ICON), which addresses the limitations of traditional methods in determining the optimum number of clusters in K-means clustering and improving the

reliability of data analysis. The contributions of the proposed algorithm can be summarized as follows:

Overcoming intuitive criteria for cluster number determination: Traditional methods often rely on intuitive criteria, such as the Elbow Method, to determine the optimal number of clusters. However, these criteria can be subjective and prone to errors. The RUN-ICON algorithm avoids such intuitive decisions and offers a systematic approach to determining the optimum number of clusters, reducing reliance on subjective judgments.

Alleviating uncertainty in clustering: Uncertainty in clustering can lead to incorrect conclusions about the underlying data structure. The RUN-ICON algorithm aims to alleviate this uncertainty by introducing new metrics, such as the Clustering Dominance Index and Uncertainty, that improve the accuracy and reliability of data analysis. The algorithm provides more reliable and informative clustering results by reducing uncertainty in K-means clustering.

Identification of stable clusters: The proposed algorithm focuses on identifying the most stable clusters that represent the true underlying structure of the data. By prioritizing stability, the algorithm ensures that the selected cluster centres accurately capture the underlying patterns and minimize the impact of outliers or noisy data points. This helps in generating more robust and interpretable clustering results.

Efficient computational performance: The RUN-ICON algorithm is designed to exhibit efficient computational performance while maintaining the accuracy and reliability of clustering results. This efficiency makes the algorithm suitable for practical applications.

Through empirical evaluations and experiments on various data sets, the study aims to demonstrate the RUN-ICON algorithm's effectiveness in overcoming traditional methods' limitations. By providing a systematic approach for determining the optimal number of clusters and improving the reliability of clustering results, the proposed algorithm offers a valuable contribution to data analysis and unsupervised learning.

The paper is organised as follows. Section 2 gives an overview of different variants of the K-means algorithm. Section 3 presents the new algorithm. Section 4 presents the benchmark problems and computational results. The conclusions from the present study are summarised in Section 5.

2. Materials and Methods

As mentioned, the K-means clustering algorithm is one of the most popular algorithms in UL. Although it was developed more than 50 years ago [19], it is still widely used today [20]. Its popularity is based on the speed at which calculations are performed [21]. Several variants of the classic K-means algorithm have been developed, each with strengths and limitations:

1. K-means++ is an improvement over the classic K-means algorithm, which addresses the issue of the initial centroid selection. K-means++ initializes the centroids by selecting the first centroid randomly and then selecting the remaining centroids with a probability proportional to the distance from the previously selected centroids [22].
2. Mini-batch K-means is a faster K-means variant that uses random data set subsets to update the centroids at each iteration. The algorithm selects a random subset of the data, assigns the data points to their nearest centroid, and then updates the centroids based on the mean of the assigned data points. The process is repeated until convergence or a defined maximum number of iterations is reached [23].
3. Online K-means is a variant of k-means that updates the centroids online, meaning that new data points can be added to the data set at any time. The algorithm starts by initializing the centroids and then updates them as new data points are added to the data set. Online K-means is useful when the data set is too large to fit into memory or when new data continuously arrives [24].
4. Kernel K-means is a variant of K-means that uses a kernel function to map the data points into a high-dimensional space. The algorithm then performs K-means clustering in the high-dimensional space, and the resulting clusters are projected

back into the original space. Kernel K-means can handle non-linearly separable data. Furthermore, it is useful in scenarios where the data cannot be separated by a linear boundary [25].

5. Spectral clustering is a variant of K-means that uses the eigenvectors of the graph Laplacian matrix to partition the data set into K clusters. The algorithm constructs a graph from the data points, where the nodes represent the data points and the edges represent their similarity. Spectral clustering is useful in scenarios where the data points have a complex structure and cannot be separated by a linear boundary [26].

3. Proposed Method

The Reduced UNcertainty-Increased CONfidence (RUN-ICON) algorithm is developed to avoid deciding on the optimum number of clusters based only on intuitive criteria, e.g., the Elbow Method. The Elbow Method [27] determines the optimal number of clusters in a data set by calculating the Within-Cluster Sum of Squares (WCSS) for different numbers of clusters. By plotting the WCSS values against the number of clusters, the elbow point, where the decrease in WCSS levels off, indicates the optimal number of clusters. The method is heuristic and requires domain knowledge for final decision-making. We aim to systematically determine the optimum number of clusters, thus ensuring prediction accuracy and relevance to the underlying data structure and identifying the most stable clusters. For this reason, we will introduce two new parameters, the Generalised Cluster Coordinate (GCC) and the Cluster Dominance Index (CDI). GCC is a vector with as many coordinates as the dimensions of the data set. Each coordinate represents a generalised coordinate of the centres of the specific dimension of a clustering event. Its derivation will be detailed later in this section. CDI is linked with the frequency of occurrence of a specific clustering configuration when requesting the splitting of the data set in a certain number of clusters and could be translated as the probability of that specific configuration occurring.

RUN-ICON utilises the essence of repeat K-means [28] and ensemble K-means [29] by repeating the clustering with random centre initialisation for any given number of clusters. However, it does not rely on the Sum of Squared Errors (SSE) or existing techniques to determine the optimum number of clusters and identify the most stable ones. The Silhouette Coefficient, for instance, is a metric often used to determine the optimum number of clusters. However, its computation highly depends on the data distribution and separation [30]. A method often used is the Jaccard similarity index. It is computationally intensive, as it requires calculating the intersection and union of all the elements in the sets [31]. Moreover, Jaccard similarity does not consider the frequency of occurrence or the order or sequence of elements in a set, which may be important for certain applications [32].

RUN-ICON selection has undergone automation, eliminating the reliance on intuitive decision-making, such as the Elbow method. Rather, it is based on CDI, i.e., the frequency of occurrence of a specific clustering configuration when requesting a particular number of clusters. CDI is calculated as:

$$CDI = \frac{\text{number of appearances of a clustering configuration}}{N} \quad (1)$$

where N represents the total number of times a clustering event was repeated for a specific number of clusters. The algorithm encompasses the following steps:

1. Run the K-means++ clustering algorithm 100 times, requesting a certain number of clusters.
2. For each run, calculate the coordinates of the centres of each cluster.
3. Compare the coordinates of the centres across all 100 runs to identify the ones that appear most frequently.
4. Choose the centres that appear most frequently as the dominant centres for this number of clusters and calculate CDI.
5. Repeat the above steps (steps 1–4) nine more times to obtain 10 sets of dominant centres.

6. Average the number of appearances of the dominant centres and the respective CDIs, obtained from the ten repetitions (ensuring that they refer to the same clustering centres). Calculate their upper and lower bounds and the variance within these two bounds.
7. Repeat steps 1–6 for different numbers of clusters. We start from 3 and run for up to 10 clusters.
8. Identify the number of clusters with the highest average CDI for all clustering possibilities (i.e., splitting from 3 and up to 10 clusters) and locate the respective dominant cluster centres. Verify if the variance in bounds is low (i.e., less than 30%)
9. Choose the number of clusters with the highest average CDI and low variance as the optimal number of clusters for the RUN-ICON algorithm.

The pseudo-code for the algorithm is given in Appendix A. Below, the part of the algorithm that compares the cluster centres through introducing the GCC is described. It is obvious that due to the random centre initialization, clusters may have the same centres but with different ordering of appearance. For example, we offer a case of 3 clusters, whereupon one initialisation, the three centres could be labelled **A**, **B**, and **C**. Then, upon a different initialization, the same centres might appear again but in a different order, e.g., **B**, **C**, **A**. Both cases should be considered as giving the same clustering. We have already discussed the shortcoming of the Jaccard similarity index to differentiate between different cluster centres. The algorithm we propose here is very simple and overcomes all shortcomings of the Jaccard index. Once all centres have been calculated, all their respective coordinates are added, thus formulating one coordinate in each data dimension, i.e.:

$$X_i = \sum_{j=1}^k (\mathbf{x}_i)_j \quad (2)$$

where X_i is the generalized coordinate of dimension- i of the data and $(\mathbf{x}_i)_j$ is the dimension- i coordinate of the j -th centre of one of the k clusters, we instruct the method to cluster the data into. Thus, for N different K-means clustering runs, each run will have only one generalised centre—regardless of the order of appearance of each cluster centre—with as many coordinates X_i as the data dimensions. In this way, it becomes easier to compare clustering frequency between different K-means runs simply by comparing their respective Generalised Clustering Coordinates (GCCs):

$$|(X_i)_l - (X_i)_m| \leq \epsilon \quad (3)$$

where $(X_i)_l$, $(X_i)_m$ are the generalised coordinates of the i dimension of clustering events l and m and ϵ is a user-defined tolerance.

The proposed algorithm incorporates the K-means++ method for initialising the centroids of the clusters. It allows a maximum of 100 iterations for the algorithm to converge, with a predefined convergence threshold set at 10^{-5} . The algorithm systematically explores the clustering configurations by iterating between 3 and 10 clusters to identify the optimal clustering configuration.

We examine clustering from 3 clusters and above since the probability of finding a hyperplane which splits the data into two groups approaches 1 rapidly. One important theorem, Hoeffding's Inequality [33], is utilised to show this. Hoeffding's Inequality is a fundamental result in probability theory that provides a bound on the probability that the sum of independent random variables deviates from its expected value by more than a certain amount. Let V_1, V_2, \dots, V_n be a sequence of independent random variables, with each variable taking values in the range $[a, b]$, and a mean of $\mu = \frac{1}{n} \sum_{i=1}^n V_i$. These n variables constitute a larger data set of N -independent variables with an expected mean M . Then, Hoeffding's Inequality states that :

$$P(|\mu - M| \geq t) \leq 2e^{\frac{-2nt^2}{(b-a)^2}} \quad (4)$$

where P is the probability and t is a non-negative parameter that determines the size of the deviation from the expected mean. This relation leads to:

$$P(|\mu - M| < t) \geq 1 - 2e^{\frac{-2nt^2}{(b-a)^2}} \quad (5)$$

Now, let us consider the problem of clustering data into two groups using a hyperplane. In this case, we can think of each data point as a random variable that takes a value of +1 if it belongs to one group and −1 if it belongs to the other group. This means that the data points are distributed within a hypercube of side length 2. Then, one hyperplane must exist, which splits the data in such a way that no matter what sample of data is randomly chosen (suppose we choose n out of the total N data points), the error during the n -data separation into two groups (empirical error) approaches the error of the total N -data separation into two groups (true error). This indicates good performance on the data set, with the hyperplane separating the data points of different classes as accurately as possible [34,35]. Hence, we need to find a condition that minimises the absolute value of the difference between these two errors. The empirical error in the n -data set may be expressed as the fraction of the misclassified values, i.e.:

$$E_n = \frac{\text{misclassified values in } n\text{-sample}}{n} \quad (6)$$

The true error for the N -data set may also be expressed as a fraction of the miscalculated points:

$$E_N = \frac{\text{misclassified values in } N\text{-sample}}{N} \quad (7)$$

This error may be broken down into two terms, one involving the misclassified points within the n -sample and another involving the misclassified points outside the n -sample:

$$E_N = \frac{\text{misclassified values within } n\text{-sample}}{N} + \frac{\text{misclassified values outside } n\text{-sample}}{N} \quad (8)$$

Since $n \leq N$, the first term in the right-hand side of Equation (8) is upper-bounded by the empirical error. Moreover, the misclassified values outside the n -sample are at most $(N - n)$. Then, Equation (8) yields:

$$E_N \leq E_n + \frac{N - n}{N} \quad (9)$$

By rearranging, and because E_N , E_n and $\frac{N-n}{N}$ are all positive numbers, we take:

$$E_n - E_N \geq -\frac{N - n}{N} \Rightarrow |E_n - E_N| \leq \frac{N - n}{N} \quad (10)$$

Applying Hoeffding's Inequality to obtain the probability of the absolute difference in errors being bounded by something small (in this case, t of Equation (5) is equal to $\frac{N-n}{N}$) yields:

$$P(|E_n - E_N| \leq \frac{N - n}{N}) \geq 1 - 2e^{\frac{-2n \frac{(N-n)^2}{N^2}}{2^2}} \quad (11)$$

where it is assumed that the probability of $|E_n - E_N|$ being exactly equal to $\frac{N-n}{N}$ is extremely small; thus, this possibility is discarded, so that $P(|E_n - E_N| \leq \frac{N-n}{N}) \approx P(|E_n - E_N| < \frac{N-n}{N})$.

$\frac{N-n}{N}$). The exponent of the second term of the right-hand side of Equation (11) may be written as:

$$\frac{-2n \frac{(N-n)^2}{N^2}}{2^2} = -\frac{n}{2} \frac{N^2 - 2Nn + n^2}{N^2} = -\frac{n^3}{2N^2} + \frac{n^2}{N} - \frac{n}{2} \quad (12)$$

In Equation (12), the numerators are consistently one order of magnitude higher than the denominators. Therefore, the exponent is influenced primarily by the numerators. The highest power of n is 3, indicating that as the number of data points n increases, the second term on the right-hand side of Equation (11) rapidly approaches 0. Consequently, since the condition of the error difference minimisation is satisfied, we conclude that the probability of discovering a hyperplane capable of separating the given data into two groups approaches 1 quickly.

Based on the above discussion and proof, separating a data set into two clusters is almost certainly feasible, and we do not need to account for this possibility when searching for optimal clustering. Moreover, it will be demonstrated that the algorithm is suitable for multi-modal data without further modifications. In the following section, we provide details about the data sets on which the RUN-ICON algorithm was tested.

4. Application of the Proposed Method

4.1. Benchmarks

To ensure a comprehensive evaluation of the algorithm, we utilised six two-dimensional artificial data sets and one multi-modal real data set, each emphasizing specific characteristics relevant to real-world data scenarios, i.e., data sets with ill-defined boundaries, data sets containing objects with distinct shapes, data sets with a high number of clusters, and data sets containing background noise. Below, we give details for the six artificial two-dimensional data sets [36] and references therein:

- Data set (a) (“2d-10c” in [36]), which contains a significantly higher number of clusters than the average data set (9 clusters), was chosen to assess the algorithm’s scalability and ability to handle data sets with numerous clusters.
- Data sets (b) (“2d-3c-no123” in [36]), (c) (“2d-4c-no4” in [36]) and (d) (“2sp2glob” in [36]) contain distinct objects either with regular or irregular shapes and were chosen to examine the algorithm’s ability to handle diverse data patterns.
- Data set (e) (“zelnik2” in [36]) contains randomly generated background noise and was chosen to test the efficiency of the algorithm in situations where noise is present.
- Data set (f) (“square2” in [36]) contains shapes with fuzzy boundaries. This data set helps evaluate the algorithm’s efficiency and effectiveness in scenarios with complex and densely distributed clusters.

All six data sets are presented in Figure 1. In Figures 1–4, the labels (a–f) refer to the respective data sets. Finally, to evaluate the algorithm’s efficiency in dealing with multi-dimensional data, a multi-modal data set was chosen and will be presented separately. This data set [37] contains data from three different varieties of wheat, which are separated based on seven different properties of each variety kernel. Due to its multi-modal nature, the visualisation of this data set is difficult.

All data sets were normalised before their introduction in the RUN-ICON algorithm to make them scale-invariant and ensure that the scale of different features does not affect the learning algorithm. This is particularly important for the multi-modal data set, where features have different scales, and those with larger values may dominate the learning process.

In the following section, the results from all test runs will be presented and discussed, and the ability of the algorithm to accurately cluster the given data sets will be assessed.

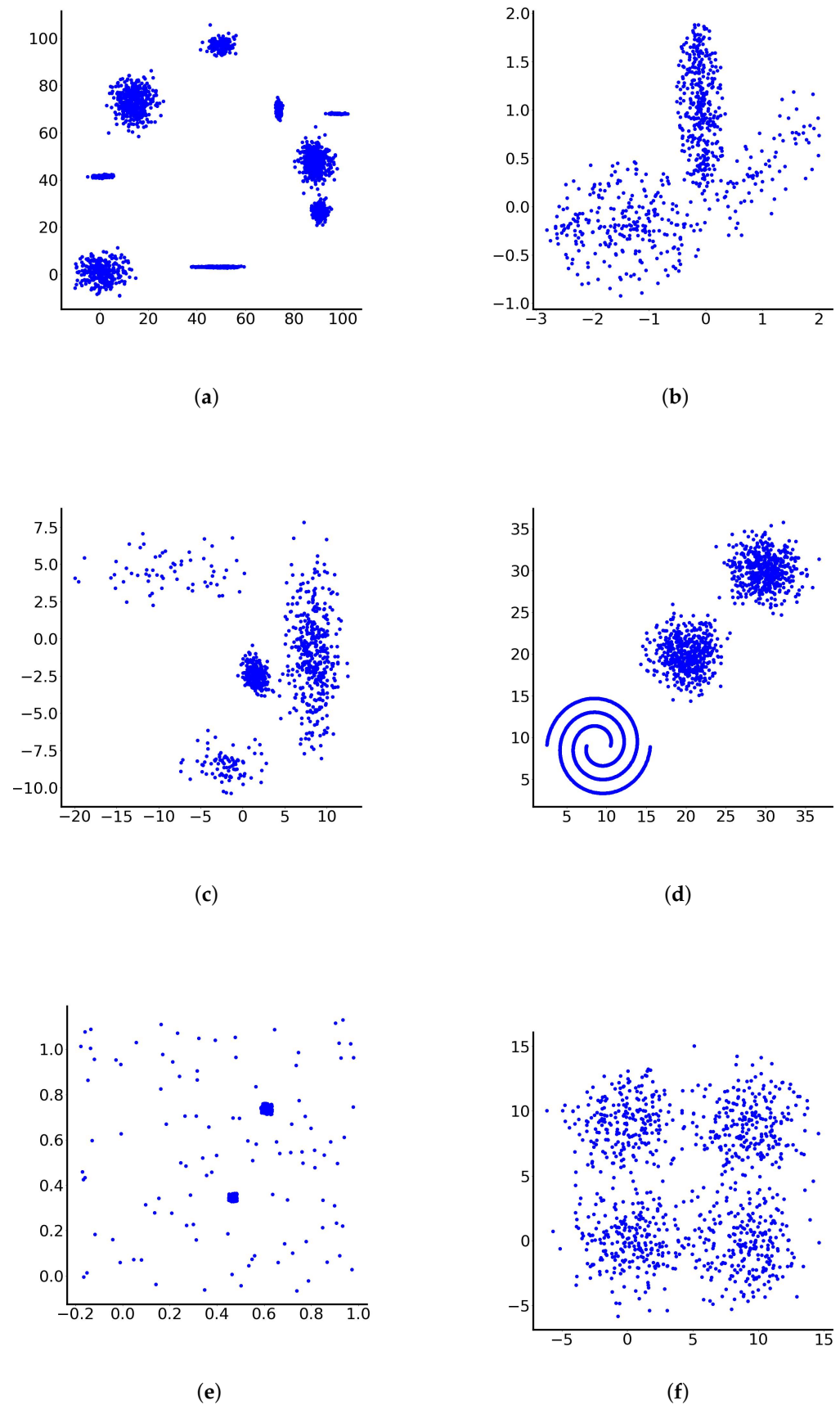
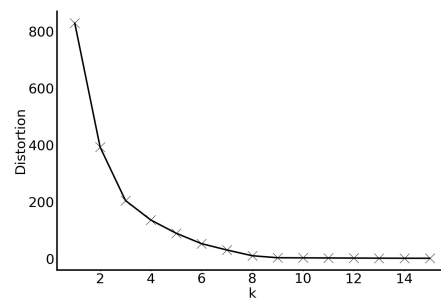
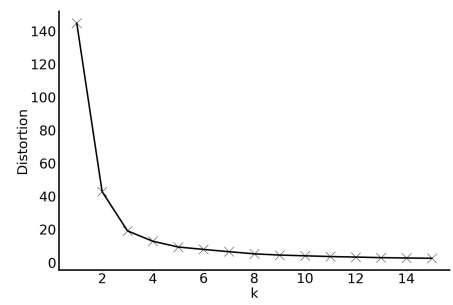


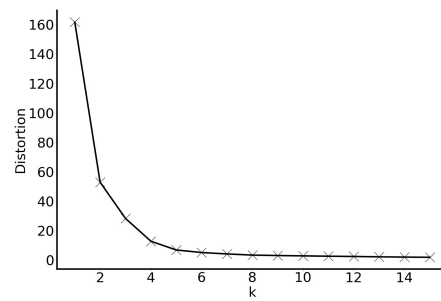
Figure 1. Presentation of the data sets for this study's six two-dimensional benchmark cases. The subfigures (a–f) correspond to the six data sets (Section 4), respectively.



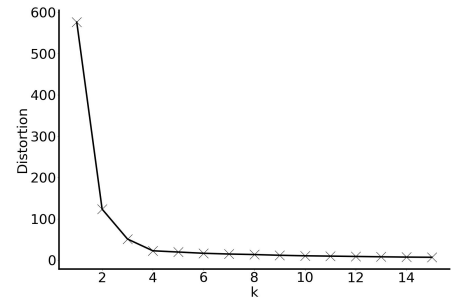
(a)



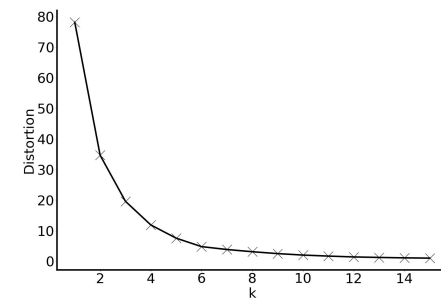
(b)



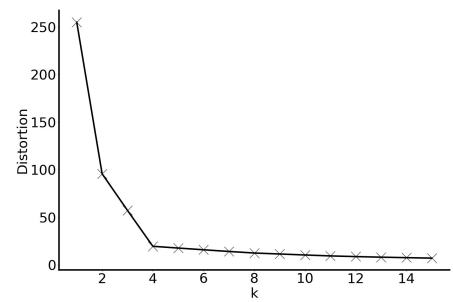
(c)



(d)



(e)



(f)

Figure 2. Results of the Elbow method for the six data sets. The subfigures (a–f) correspond to the 6 data sets (Section 4), respectively.

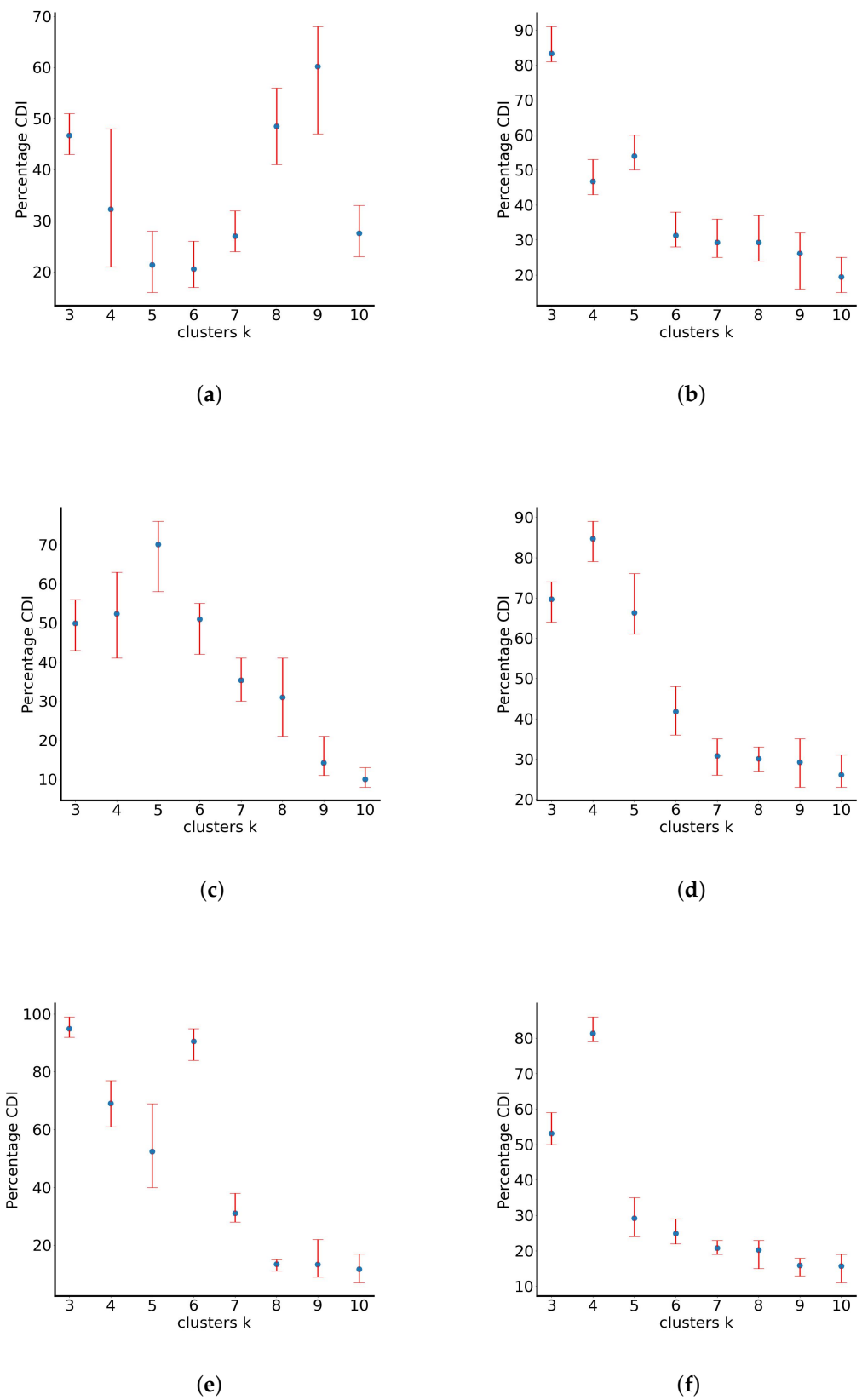
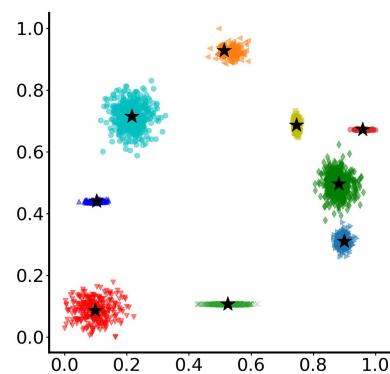
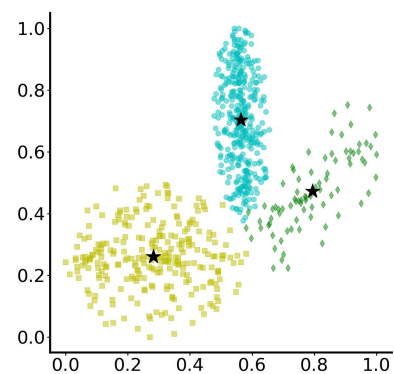


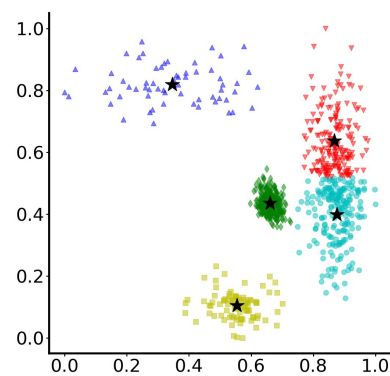
Figure 3. Dominant clustering results (averaged percentage CDIs) for different clustering requirements ranging from 3 to 10 clusters. The subfigures (a–f) correspond to the 6 data sets (Section 4), respectively.



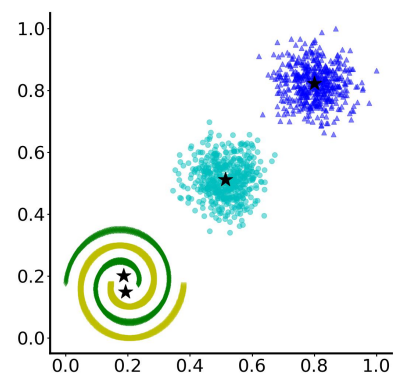
(a)



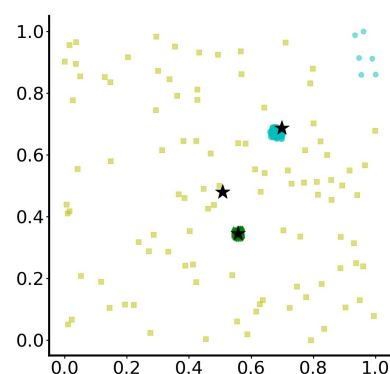
(b)



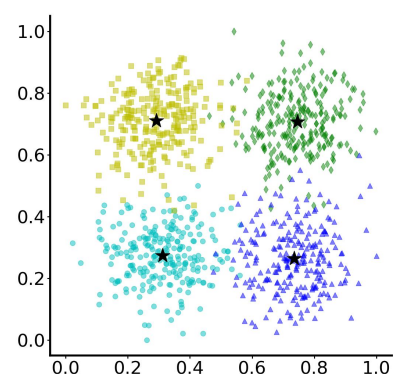
(c)



(d)



(e)



(f)

Figure 4. Optimal clusters generated by the RUN-ICON algorithm. Different colours correspond to different clusters. The black stars correspond to the cluster centres. The subfigures (a–f) correspond to the 6 data sets (Section 4), respectively.

4.2. Two-Dimensional Data Sets

In Figure 2, the results of the Elbow method are presented for all six data sets, utilising K-means++. It is not straightforward to conclude the optimal number of clusters for each data set. Since the only criterion for the method is Euclidean distance minimisation, it is only natural for the error to reduce as the number of clusters increases, and the distances between cluster centres and cluster points reduce. For data set (a), for example, by observing the Elbow graph, there is no possibility of deducing that the optimal number of clusters is 9. Probably, only for data set (f) can one deduce from the graph with some certainty that the number of clusters is four since there is a clear break in the graph for four clusters. However, one can not be certain if rerunning the algorithm will lead to the same clustering configuration or a different one, which might have a different error. Hence, a different elbow point might appear on the graph.

In contrast, when the RUN-ICON was applied, the results demonstrated a clear dominance of a specific clustering pattern. Figure 3 illustrates the dominant clustering outcomes (averaged percentage CDIs) for different clustering requirements (from three to 10 clusters), as calculated by step 6 of the algorithm described in Section 3. These are depicted along with their respective upper and lower bounds. The bigger difference in averaged CDIs between the dominant clustering and the rest leads to Increased CONFidence (ICON) that the correct number of optimal clusters has been calculated. Taking into account the upper and lower bounds for a clustering configuration, we may introduce the uncertainty equation:

$$Uncertainty = \frac{\text{upper bound} - \text{lower bound}}{\text{upper bound}} \quad (13)$$

The smaller variance between upper and lower bounds for the proposed optimal clustering leads to Reduced UNCertainty (RUN) in the selection process. The combination of the two, where CDI is highest and uncertainty is low (an upper bound of 30% has been set in the algorithm), indicates whether a particular clustering is optimal for that data set. The separation into two clusters has been omitted. Still, it must be mentioned that RUN-ICON confirmed that the probability of splitting a data set into two clusters is very high, giving average CDIs ranging from 0.73 to 1.0 for all six data sets. These CDIs were indeed the maximum values for their respective data sets.

The optimal clustering proposed by the researchers who generated the data sets are nine clusters for data set (a); three clusters for data set (b); four clusters for data set (c); four clusters for data set (d); three clusters for data set (e); four clusters for data set (f). These and a depiction of these clusters may be found in [36]. The optimal clustering and the respective average highest CDI and uncertainty for all datasets are presented in Table 1.

Table 1. Performance of the RUN-ICON algorithm.

Data Set	Predicted Clusters	Average Highest CDI	Uncertainty
(a)	9	0.62	0.30
(b)	3	0.83	0.10
(c)	5	0.70	0.23
(d)	4	0.85	0.11
(e)	3	0.95	0.07
(f)	4	0.81	0.08

The clusters generated by RUN-ICON for the normalised data sets are presented in Figure 4, and we examine their similarities to the intended clusters. The only incorrect clustering occurs for data set (c), where the largest cluster is split into two smaller clusters. Thus, the algorithm predicts five clusters instead of the proposed four. This could be

attributed to the algorithm's attempt to ensure compatibility with the size of the remaining clusters in the data set. For the other five data sets, the correct number of optimal clusters was predicted. An interesting outcome is that for data set (e) (the one with the background noise), six outliers from the "noise" were clustered together with the topmost cluster of concentrated data. This could be because outliers sometimes possess unique or distinctive features that set them apart from other data points. The algorithm could have captured these features in the same cluster as the concentrated data. The algorithm may have identified shared characteristics or patterns that justified their inclusion, even if they were not physically close. It is important to note that the specific behaviour and reasoning behind the algorithm's clustering decisions may depend on the algorithm itself, its parameter settings, and the nature of the data.

Furthermore, a comparison was made between the optimal clustering predicted by RUN-ICON, standard K-means++, Ward hierarchical clustering, Bayesian K-means and repeat K-means algorithms (with the Silhouette Coefficient being utilised as a measure of performance for the other algorithms). As already discussed, our algorithm correctly matched the optimal clustering for five out of the six data sets (calculating only for data set (c) one additional cluster). The standard K-means++, Ward, and repeat K-means algorithms exhibited similar behaviour. They all missed the distinction in the two circular shapes of the data set (d), clustering them together in one cluster. They also missed the background noise in the data set (e). K-means++ proposed six or 10 clusters as the optimal number for that data set. Ward and repeat K-means algorithms proposed 10 clusters as the optimal number for that data set. Bayesian K-means algorithm predicted one extra cluster in the data set (c) (five instead of four) and also missed the background noise in the data set (e), proposing six clusters. The results are summarised in Table 2 (the proposed optimal clustering is also given for comparison), along with the average prediction error, which is defined as follows:

$$Error = \frac{1}{6} \sum_{i=1}^6 \frac{|\text{predicted optimal clusters} - \text{intended optimal clusters}|}{\text{intended optimal clusters}} \quad (14)$$

where six refers to the total number of data sets.

Table 2. Comparisons between the RUN-ICON and other UL algorithms. PONC stands for Proposed Optimum Number of Clusters.

Data Set	PONC	RUN-ICON	K-m++	Ward	Bayesian K-m	Repeat K-m
(a)	9	9	9	9	9	9
(b)	3	3	3	3	3	3
(c)	4	5	4	4	5	4
(d)	4	4	3	3	4	3
(e)	3	3	6 or 10	10	6	10
(f)	4	4	4	4	4	4
Error (%)		3	21 or 43	43	21	43

As observed, the overall prediction error for RUN-ICON is of the order of 3%, which is by far the smallest of all, with the algorithm being able to predict correctly five out of six optimal data set clusterings. All other methods predicted four out of six optimal clusterings, completely missing the background noise in the data set (e). From this analysis, it becomes clear that the performance of our algorithm is notably superior in accurately predicting the number of clusters required for optimal clustering.

In addition to evaluating the clustering results, the computational time required to run the algorithm is also discussed. All computations were conducted on an 8-core 2.4 GHz Intel i5-9300HF processor with 8 Gbytes of RAM. Table 3 details the number of data points in each data set and the computational time to split each data set into three, five and

ten clusters, respectively. It appears that the computational time for each event follows a complexity of $O(h)$, considering the number of data points in each data set. From the presented results becomes clear that the algorithm demonstrates notable efficiency by executing with minimal computational time, thereby indicating its ability to perform tasks rapidly without imposing a substantial burden on computational resources.

Table 3. Computational performance of the RUN-ICON algorithm.

Data Set	Data Points	T_3 (s)	T_5 (s)	T_{10} (s)
(a)	2997	18	20	28
(b)	725	18	24	27
(c)	873	18	21	31
(d)	2014	26	33	42
(e)	314	15	19	21
(f)	1007	19	25	29

4.3. Multi-Modal Data Set

Furthermore, the performance of the RUN-ICON algorithm was examined for the multi-modal data set. The data set comprises 210 data points of three different wheat varieties, with 70 randomly selected kernels from each variety, where seven different geometric attributes are given, i.e., a seven-dimensional data set) [37].

Figure 5 presents the dominant clustering outcomes (averaged percentage CDIs) for different clustering requirements (from three to 10 clusters), as calculated by step 6 of the algorithm described in Section 3. These are depicted along with their respective upper and lower bounds. The dominant clustering predicted by RUN-ICON splits the data into three clusters, representing the three different wheat varieties described in the data set, with the averaged CDI exceeding 0.95. At the same time, uncertainty was extremely low for this choice of clustering, i.e., 0.05. The computational run times, which vary between 10 s and 18 s for splitting into three and 10 clusters, respectively, confirm the assumption of the algorithm's $O(h)$ time complexity.

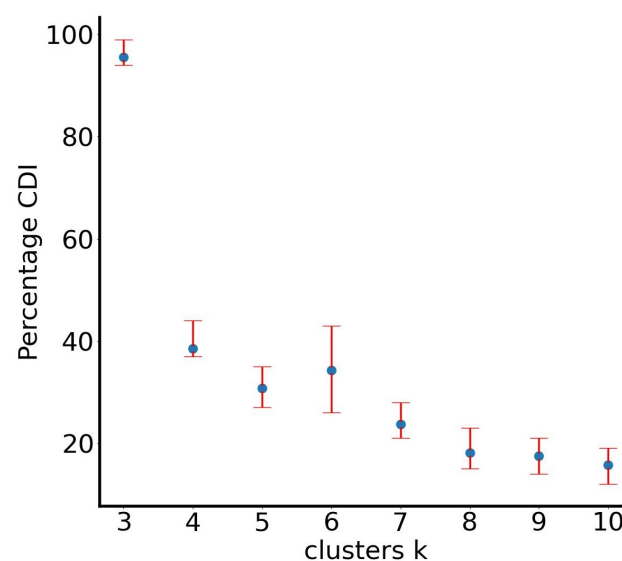


Figure 5. Percentage averaged CDI for the multi-modal data set.

5. Conclusions

This study introduces a novel unsupervised learning algorithm, RUN-ICON (Reduce UNcertainty and Increase CONfidence). The algorithm demonstrates robustness by effec-

tively determining the optimal number of clusters by identifying commonly dominant centres across multiple repetitions of the K-means++ algorithm. The algorithm does not rely on the Sum of Squared Errors and determines optimal clustering by introducing novel metrics such as the Clustering Dominance Index and Uncertainty. Extensive numerical tests were conducted on diverse data sets to evaluate the algorithm's performance. The results highlight its exceptional accuracy in identifying the correct number of optimal clusters under various scenarios. RUN-ICON offers several advantages, including improved robustness, enhanced clustering quality, automation, and flexibility.

Firstly, the algorithm's robustness stems from its utilization of the frequency of appearance of dominant centres over multiple repetitions. This approach diminishes the influence of outliers and random variations in the initial clustering process. Consequently, the algorithm becomes more resilient to noise and fluctuations in the data, resulting in more reliable clustering outcomes.

Moreover, the algorithm's capacity to select the number of clusters based on the most frequently occurring dominant centres enhances the overall quality of the clustering results. By prioritizing the clusters with greater representation, the algorithm increases the likelihood of producing meaningful and insightful clusters for subsequent analysis.

Furthermore, the algorithm's fully automated nature contributes to its efficiency and eliminates potential biases in selecting the optimal number of clusters. The algorithm saves time and enhances reproducibility by removing the need for manual intervention or subjective decisions.

Additionally, the algorithm exhibits remarkable flexibility as it can be applied to various data sets without relying on specific assumptions about the underlying data distribution. This versatility allows the algorithm to be employed in diverse problem domains, accommodating data characteristics and structures.

One potential limitation of the algorithm is its dependency on repetitions: RUN-ICON relies on multiple repetitions of the K-means++ algorithm to identify commonly dominant centres. While this approach enhances robustness, it increases the computational complexity and time required. To mitigate this limitation, one potential remedy is to investigate adaptive or dynamic strategies for determining the number of repetitions based on convergence or stability criteria, reducing unnecessary repetitions without sacrificing accuracy.

Future work will focus on applying the RUN-ICON algorithm to tackle a wide range of problems and data sets in the framework of both unsupervised and deep learning [38], where uncertainties also exist during data analysis, especially in the case of sparse and noisy data. This will further validate its effectiveness and explore its applicability in various real-world scenarios, extending its practical utility in diverse fields of study.

Author Contributions: Conceptualization, N.C. and D.D.; methodology, N.C. and D.D.; formal analysis, N.C. and D.D.; investigation, N.C. and D.D.; resources, D.D.; writing, N.C. and D.D.; project administration, D.D.; funding acquisition, D.D.; contribution to the discussion N.C. and D.D. All authors have read and agreed to the published version of the manuscript.

Funding: This paper is supported by the European Union's Horizon Europe Research and Innovation Actions programme under grant agreement No 101069937, project name: HS4U (HEALTHY SHIP 4U). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Climate, Infrastructure, and Environment Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. The Algorithm

Algorithm A1 RUN-ICON

```

for each num_clusters from 3 to 10 do
  highest_avg_cdi = 0
  dominant_centers = [ ]
  for repetition in range(10) do
    center_frequency = { }
    cluster_centers = [ ]
    for run in range(100) do
      initialize cluster_centers using K-means++ algorithm with num_clusters
      calculate coordinates of centres for each cluster in cluster_centers
      if coordinates not in center_frequency then
        center_frequency[coordinates] = 1
      else
        center_frequency[coordinates] = center_frequency[coordinates] + 1
      end if
    end for
    most_frequent_centers = get_most_frequent_centers(center_frequency)
    cdi = calculate_cdi(most_frequent_centers)
  end for
  avg_cdi = average(cdi) overall repetition indices
  calculate variance from upper and lower bounds of avg_cdi
  if avg_cdi > highest_avg_cdi and variance ≤ 30% then
    highest_avg_cdi = avg_cdi
    dominant_centers = most_frequent_centers
    optimal_num_clusters = each_num_clusters
    store dominant_centers
  end if
end for

```

References

1. Hinton, G.E.; Dayan, P.; Frey, B.J.; Neal, R.M. The “Wake-Sleep” Algorithm for Unsupervised Neural Networks. *Science* **1995**, *268*, 1158–1161.
2. Krotov, D.; Hopfield, J.J. Unsupervised learning by competing hidden units. *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 7723–7731.
3. Hadsell, R.; Chopra, S.; LeCun, Y. Dimensionality reduction by learning an invariant mapping. In Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), New York, NY, USA, 17–22 June 2006; IEEE: Piscataway, NJ, USA, 2006; Volume 2, pp. 1735–1742.
4. Alloghani, M.; Al-Jumeily Obe, D.; Mustafina, J.; Hussain, A.; Aljaaf, A. A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science. In *Supervised and Unsupervised Learning for Data Science*; Springer: Cham, Switzerland, 2020; pp. 3–21. [\[CrossRef\]](#)
5. Na, S.; Xumin, L.; Yong, G. Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm. In Proceedings of the 2010 Third International Symposium on Intelligent Information Technology and Security Informatics, Jian, China, 2–4 April 2010; pp. 63–67. [\[CrossRef\]](#)
6. Murtagh, F.; Contreras, P. Algorithms for hierarchical clustering: An overview. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2012**, *2*, 86–97. [\[CrossRef\]](#)
7. Lee, J.; Lee, G. Feature Alignment by Uncertainty and Self-Training for Source-Free Unsupervised Domain Adaptation. *Neural Netw.* **2023**, *161*, 682–692. [\[CrossRef\]](#)
8. Lee, J.; Lee, G. Unsupervised domain adaptation based on the predictive uncertainty of models. *Neurocomputing* **2023**, *520*, 183–193. [\[CrossRef\]](#)
9. Mousavi, Z.; Yousefi Rezaii, T.; Sheykhivand, S.; Farzamnia, A.; Razavi, S. Deep convolutional neural network for classification of sleep stages from single-channel EEG signals. *J. Neurosci. Methods* **2019**, *324*, 108312. [\[CrossRef\]](#)
10. Mousavi, Z.; Varahram, S.; Mohammad Eftefagh, M.; Sadeghi, M.H. Dictionary learning-based damage detection under varying environmental conditions using only vibration responses of numerical model and real intact State: Verification on an experimental offshore jacket model. *Mech. Syst. Signal Process.* **2023**, *182*, 109567. [\[CrossRef\]](#)
11. Orosz, T.; Vagi, R.; Mark, C.; Nagy, D.; Vadasz, P.; Istvan, A.; Megyeri, A. Evaluating Human versus Machine Learning Performance in a LegalTech Problem. *Appl. Sci.* **2021**, *12*, 297. [\[CrossRef\]](#)

12. Melnykov, V.; Michael, S. Clustering Large Datasets by Merging K-Means Solutions. *J. Classif.* **2019**, *37*, 97–123. [\[CrossRef\]](#)
13. Vellido, A. The importance of interpretability and visualization in machine learning for applications in medicine and health care. *Neural Comput. Appl.* **2020**, *32*, 18069–18083. [\[CrossRef\]](#)
14. Pintelas, E.; Livieris, I.; Pintelas, P. A Convolutional Autoencoder Topology for Classification in High-Dimensional Noisy Image Datasets. *Sensors* **2021**, *21*, 7731. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006.
16. Zou, Z.; Hua, K.; Zhang, X. HGC: Fast hierarchical clustering for large-scale single-cell data. *Bioinformatics* **2021**, *37*, 3964–3965. [\[CrossRef\]](#) [\[PubMed\]](#)
17. Mehra, A.; Kailkhura, B.; Chen, P.Y.; Hamm, J. Understanding the Limits of Unsupervised Domain Adaptation via Data Poisoning. In *Advances in Neural Information Processing Systems*; Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2021; Volume 34, pp. 17347–17359.
18. Frank, M.; Drikakis, D.; Charissis, V. Machine-Learning Methods for Computational Science and Engineering. *Computation* **2020**, *8*, 15. [\[CrossRef\]](#)
19. MacQueen, J. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, CA, USA, 1967; pp. 281–297.
20. Ahmed, M.; Seraj, R.; Islam, S.M.S. The k-means Algorithm: A Comprehensive Survey and Performance Evaluation. *Electronics* **2020**, *9*, 1295. [\[CrossRef\]](#)
21. Har-Peled, S.; Sadri, B. How Fast Is the k-Means Method? *Algorithmica* **2005**, *41*, 185–202. [\[CrossRef\]](#)
22. Arthur, D.; Vassilvitskii, S. K-Means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007*, New Orleans, LA, USA, 7–9 January 2007; Volume 8, pp. 1027–1035. [\[CrossRef\]](#)
23. Sculley, D. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web*, Raleigh, NC, USA, 26–30 April 2010; pp. 1177–1178. [\[CrossRef\]](#)
24. Cohen-Addad, V.; Guedj, B.; Kanade, V.; Rom, G. Online k-means Clustering. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, Virtual, 13–15 April 2021; Volume 130, pp. 1126–1134.
25. Schölkopf, B.; Smola, A.; Müller, K.R. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Comput.* **1998**, *10*, 1299–1319. [\[CrossRef\]](#)
26. Ng, A.; Jordan, M.; Weiss, Y. On Spectral Clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*; Dietterich, T., Becker, S., Ghahramani, Z., Eds.; MIT Press: Cambridge, MA, USA, 2001; Volume 14.
27. Shi, C.; Wei, B.; Wei, S.; Wang, W.; Liu, H.; Liu, J. A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm. *EURASIP J. Wirel. Commun. Netw.* **2021**, *2021*, 31. [\[CrossRef\]](#)
28. Fränti, P.; Sieranoja, S. How much can k-means be improved by using better initialization and repeats? *Pattern Recognit.* **2019**, *93*, 95–112. [\[CrossRef\]](#)
29. Kim, E.Y.; Kim, S.Y.; Ashlock, D.; Nam, D. MULTI-K: Accurate classification of microarray subtypes using ensemble k-means clustering. *BMC Bioinform.* **2009**, *10*, 260. [\[CrossRef\]](#)
30. Shutaywi, M.; Kachouie, N. Silhouette Analysis for Performance Evaluation in Machine Learning with Applications to Clustering. *Entropy* **2021**, *23*, 759. [\[CrossRef\]](#)
31. Besta, M.; Kanakagiri, R.; Mustafa, H.; Karasikov, M.; Ratsch, G.; Hoefler, T.; Solomonik, E. Communication-Efficient Jaccard similarity for High-Performance Distributed Genome Comparisons. In *Proceedings of the 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, New Orleans, LA, USA, 18–22 May 2020; pp. 1122–1132. [\[CrossRef\]](#)
32. Manning, C.D.; Raghavan, P.; Schuetze, H. *Introduction to Information Retrieval*; Cambridge University Press: Cambridge, UK, 2008. [\[CrossRef\]](#)
33. Hoeffding, W. Probability Inequalities for sums of Bounded Random Variables. In *The Collected Works of Wassily Hoeffding*; Fisher, N.I., Sen, P.K., Eds.; Springer: New York, NY, USA, 1994; pp. 409–426. [\[CrossRef\]](#)
34. Burges, C.; Burges, C.J. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Min. Knowl. Discov.* **1998**, *2*, 121–167. [\[CrossRef\]](#)
35. Ting, D. Count-Min: Optimal Estimation and Tight Error Bounds using Empirical Error Distributions. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, London, UK, 20–23 August 2018; pp. 2319–2328. [\[CrossRef\]](#)
36. Barton Tomas. Clustering Benhmarks. 2023. Available online: <https://github.com/deric/clustering-benchmark> accessed on 4 January 2023).
37. Charytanowicz, M.; Niewczas, J.; Kulczycki, P.; Kowalski, P.; Lukasik, S. Seeds Data Set. Available online: <http://archive.ics.uci.edu/ml/datasets/seeds> (accessed on 21 May 2023).
38. Poulinakis, K.; Drikakis, D.; Kokkinakis, I.W.; Spottswood, S.M. Machine-Learning Methods on Noisy and Sparse Data. *Mathematics* **2023**, *11*, 236. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.